

Data Wrangling with SQL

Good morning !

We start at 10:00

Codd's Normal Forms (I)

The Goal :

- ① Eliminate data inconsistencies
 - ② Minimize storage space as much as possible with respect to ①
- Constraint-based optimization problem.

First Normal Form (1NF)

All attributes of a relation
columns table

should be atomic in

data information → Value → smallest,
non-decomposable

Break
of 1NF

1 NF (2)

CUSTOMERS

(CustID, (CustName, Address)

<u>CustID</u>	<u>CustName</u>	<u>Address</u>
		Street# Street name
1	JT (Ho)	10 Downing Street, London, UK
		City County

INF is
respected

INF (3)

Addresse

CUSToNERS		(CustID, CustName, St.H, St.Nm, City, County)			
CustID	CustName	St. H	St. Nm	City	County
1	TT (H0)	10	Dowhing	London	UK

Second Normal Form (2NF) (1)

- F_o, 2NF to apply:
- 1NF should apply
- + There shall be a functional dependency
between Part of the Key and a
non-key attribute

INF \rightarrow YES

2NF \rightarrow NO

2NF (2)

not-key
but-key

Player (Prison, Sport, Height, Weight)

Player	Sport	Height	Weight	B_Pack
Spiderman	run	190	70	10
Hulk	swim	170	70	10
Thor	tennis	170	70	10

$1NF \rightarrow Yes$ $2NF$ (3) $2NF$
 $2NF \rightarrow Yes$ $\overbrace{\quad\quad\quad}$ "respected"

Olympic (Person, Sport)

Person	Sport
Sebastian	run
Hannah	swim
Hanna	Technik

$2NF$
Player (Person, Height, weight)

Person	Height	Weight
Sebastian	190	90
Hanna	170	70

Third Normal Form (3NF) (1)

For 3NF to apply:

2NF should apply

+ There shall be no functional
dependency between non-key
attributes

INF → YES

2NF → YES

3NF →

3NF (2)

Breach of
3NF

VEHICLE (Reg#, Type, brand, Color, EngineP)

Reg#	Type	brand	Color	Engine
1001	Golf	WW	black	1.2
1002	Clio	Renault	white	1.2
1003	Captur	Renault	black	1.5



1NF → Yes

2NF → Yes

3NF → Yes

3NF (3)

VEHICLE(Reg#, Type, (l.o.))

FK

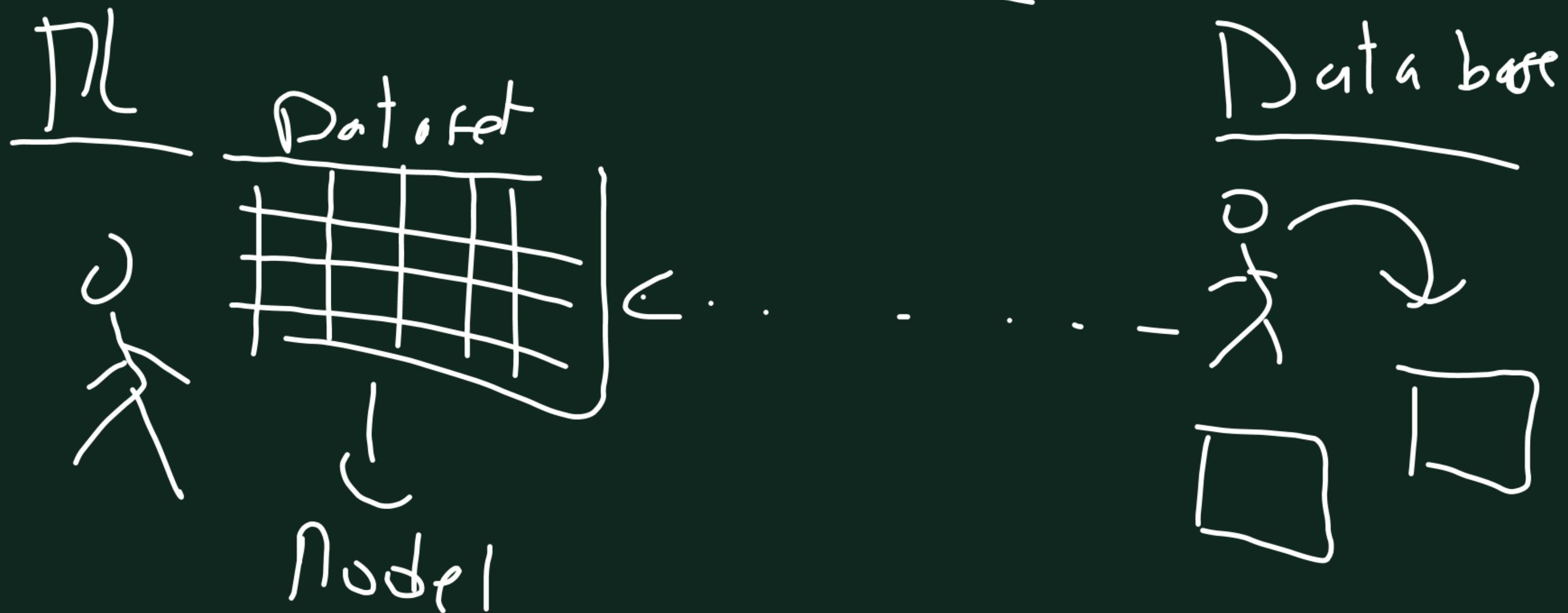
3NF is
'respected'
PK

MODEL(Type, Brand, Engine)

Reg#	Type	(l.o.)
1001	GOLF	Black
1002	Clio	White
1003	Captur	Black

Type	Brand	Engine
GOLF	VW	1.2
Clio	Renault	1.2
Captur	Renault	1.5

Chain of Data



Godd's Normal Forms (2)

Recent findings (in the 70's) showed that while respecting the NF,

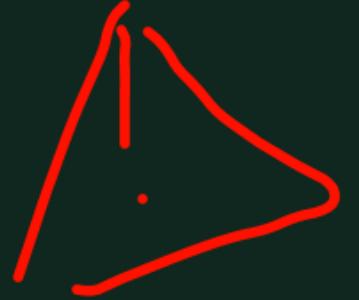
roughly 5% of db's are still inconsistent

Boyce-Codd Normal Form (BCNF)

There shall be no other
functional dependency aside from
excludes key and non-key attributes
key - key
partial key - R
key, etc.

Break

Back at 11 : 50



Normal Forms: A theoretical Perspective

T (A, B, C, D)

INF: A, B, C, D are atomic

2NF: A, B \rightarrow C

A, B \rightarrow D

Not A \rightarrow C

Not A \rightarrow D

Not B \rightarrow C

Not B \rightarrow D

3NF:

Not C \rightarrow D

Not D \rightarrow C

BCNF:

Not C \rightarrow A

Not C \rightarrow B

Not D \rightarrow A

Not D \rightarrow B

(+f, completeness: Not A \rightarrow B)

WWI example : INF

```
use WideWorldImporters
select * from Warehouse.StockItems
```

	InternalComments	Photo	CustomFields	Tags	SearchDetails
1	NULL	NULL	{"CountryOfManufacture": "China", "Tags": ["USB"]}	["USB Powered"]	USB missile launcher (Gr...
2	NULL	NULL	{"CountryOfManufacture": "China", "Tags": ["USB"]}	["USB Powered"]	USB rocket launcher (Gr...
3	wall? This is just ...	NULL	{"CountryOfManufacture": "China", "Tags": ["USB"]}	["USB Powered"]	Office cube periscope (B...
4	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["32G..."]}	["32GB", "USB Powered"]	USB food flash drive - su...
5	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["16G..."]}	["16GB", "USB Powered"]	USB food flash drive - ha...
6	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["32G..."]}	["32GB", "USB Powered"]	USB food flash drive - hc...
7	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["16G..."]}	["16GB", "USB Powered"]	USB food flash drive - pi...
8	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["32G..."]}	["32GB", "USB Powered"]	USB food flash drive - di...
9	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["16G..."]}	["16GB", "USB Powered"]	USB food flash drive - ba...
10	NULL	NULL	{"CountryOfManufacture": "Japan", "Tags": ["32G..."]}	["32GB", "USB Powered"]	USB food flash drive - ch...

{ "Country": "China",
 "Tags": ["USB",
 "Cable"] }
R-EACH
/AF, L

WWI example : 3NF

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'WideWorldImporters' database is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following code:

```
use WideWorldImporters
select * from Sales.Invoices
```

A yellow bracket labeled 'PK' is drawn around the 'CustomerID' and 'BillToCustomerID' columns in the result set. The result set displays 70,510 rows of data.

InvoiceID	CustomerID	BillToCustomerID	OrderID	DeliveryMethodID	ContactPersonID	AccountsPersonID	SalespersonPersonID	PackedByPersonID
1	852	803	1	3	3032	3032	2	14
2	803	803	2	3	3003	3003	8	14
3	105	1	3	3	1209	1001	7	14
4	57	1	4	3	1113	1001	16	14
5	905	905	5	3	3105	3105	3	14
6	976	976	6	3	3176	3176	13	14
7	575	401	7	3	2349	2001	8	14
8	964	964	8	3	3164	3164	7	14
9	77	1	9	3	1153	1001	7	14
10	191	1	10	3	1381	1001	20	14

At the bottom of the screen, the status bar shows: Status: Running | Ln 3 Col 29 Ch 29 INS ENG 3:08 AM FR 5/17/2024

no h - key
attributes
(related in
CUSTOMERS
table)
Breach of 3NF

Lunch

Back at 14:00

Seb's Playground DB

In the Purchased

+ table,
and the
line:

1	40	2024-05-05
---	----	------------

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'A22_Seb_Playground' database and its tables like 'Customers', 'Products', and 'Purchased'. The 'SQLQuery1.sql' window on the right contains the following SQL code:

```
use A22_Seb_Playground
select * from Purchased
```

The results grid displays the following data:

	CustomerID	ProductID	DateOfPurchase	Quantity
1	1	10	2023-01-01	1
2	1	10	2024-05-17	2
3	1	20	2022-12-01	2
4	1	30	2022-12-25	3
5	1	40	2024-05-05	3
6	2	10	2022-09-09	1
7	2	10	2022-10-10	1
8	2	20	2022-11-11	1

An orange arrow points to the second row of the results grid, highlighting the purchase of product ID 10 by customer ID 1 on May 17, 2024.

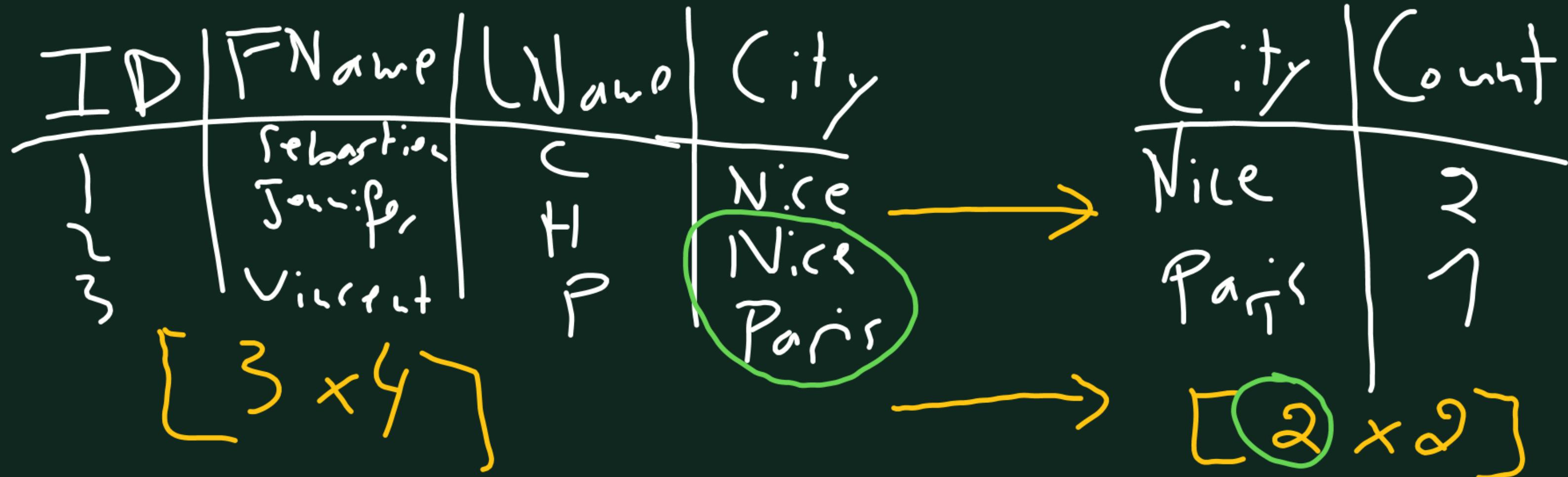
Group by Query ()

Result :

City	Count of Customers
Nice	2
Paris	1

Group by Query (2)

Dimensionality Reduction



B PRODUCTS

E/R Diagram

A C U S T O M E R S

A	B	C
ProductID	Description	UnitPrice
10	Laptop	800
20	Desktop	1000
30	Tablet	600
40	Flat Monitor	150

E	F	G	H	I	J	K	L
CustomerID	FirstName	LastName	Email	City			
1	Seb	C	sebastien@dsti.c	Nice			
2	Jennifer	H	jennifer@dsti.c	Nice			
4	Vincent	P	vincent@dsti.c	Paris			

(0, n)

(0, h)

CustomerID	ProductID	DateOfPurchase	Quantity
1	10	1/1/2023	1
1	10	5/17/2024	2
1	20	12/1/2022	2
1	30	12/25/2022	3
1	40	5/5/2024	3
2	10	9/9/2022	1
2	10	10/10/2022	1
2	20	11/11/2022	1

PURCHASED AB

Division Query (I)

→ Simplest, most effective & optimized way to solve the

Question :
(Customers) (Purchased)

Give all entityA that relationship A B

all entityB ?
(Product)

Division Query (?)

Conditions to fulfill:

- Entity A
- Entity B
- Relationship AB binding them

Division Query (3)

Skeleton:

- Outer query (entity A)
- Inner (sub) query (entity B)
- Inner inner (sub-sub) query (Relationship AB)

Division Quirk (4)

Why does it work?

I am happy → 

I am not happy → 

I am hot not happy → 

All Customers \rightarrow All Products

PRODUCT

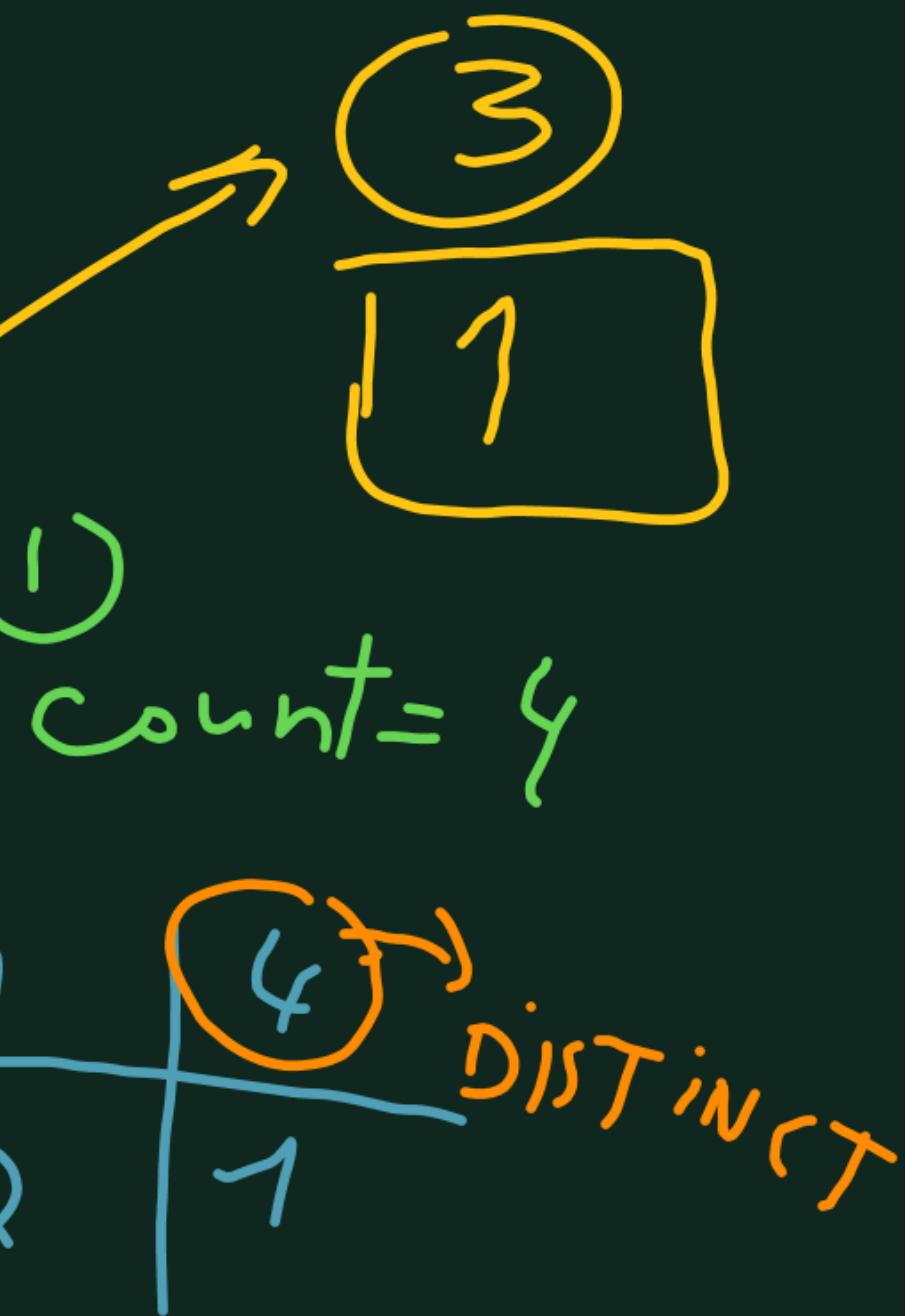
Product ID	Product Name
10	Laptop
20	Monitor
30	PC
40	Batteries

② count = 4

PURCHASED

Customer ID	Product ID
1	10
2	70
1	30
1	110
2	10
2	20

① 2



Break

Back at 15:20

Programmatic SQL (I)

What's the problem with
weaknesses/limitations

"Popular" SQL?

Programmatic SQL (?)

Aim: bridge gap between SQL
and functional languages
(e.g., Python)

Programmatic SQL (3)

Why ?

1

Locality of data

2

Intellectual Property

Programmatic SQL (4)

① Locality of data

Generally, data flows in 3

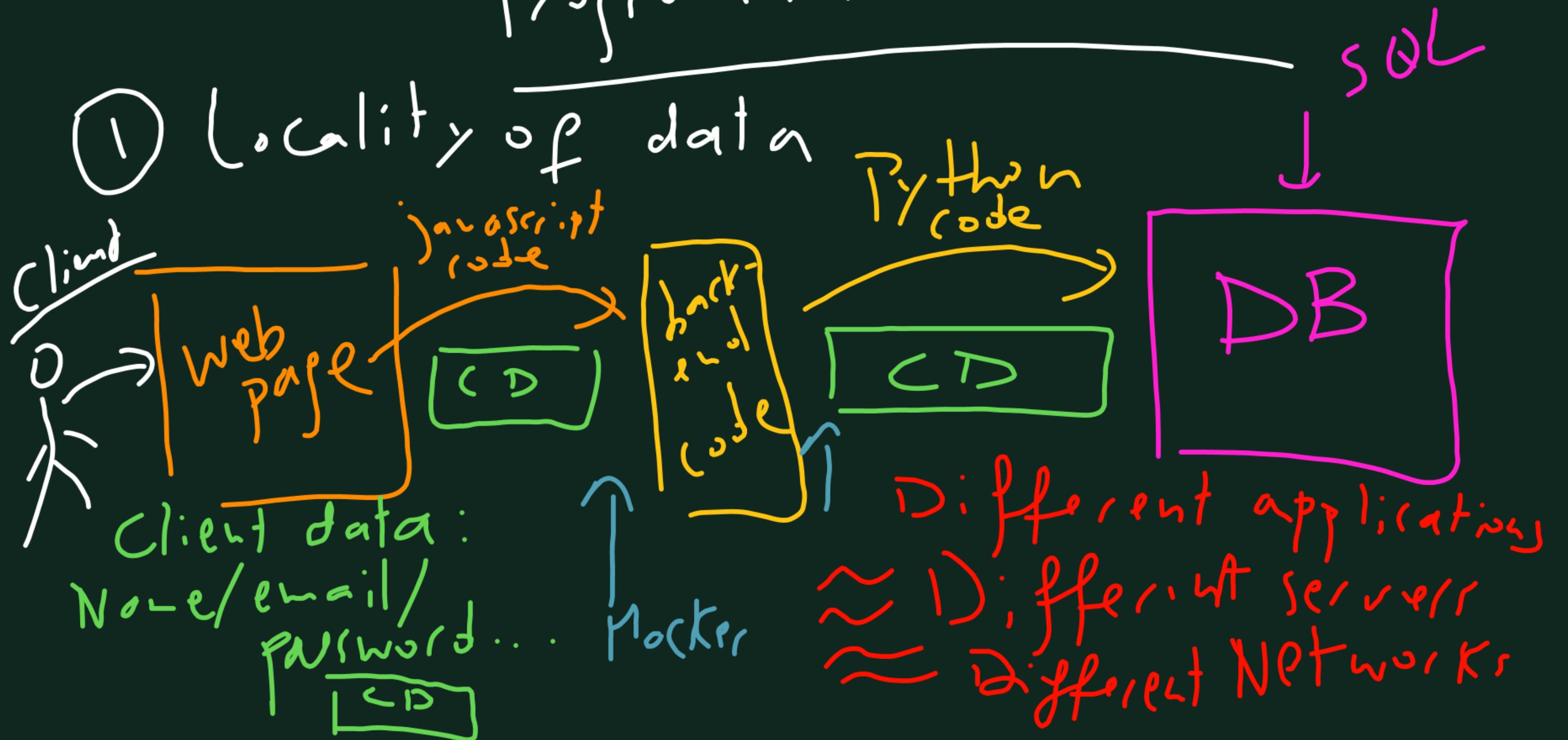
“Parts”

① UI (website)
Front-end

② Back-end

③ DB

Programmatic SQL (5)



Programmatic SQL (6)

② Intellectual Property

→ "He who owns the code,
owns the data".

Programmatic SQL (?)

- Benefits: (loop, define variables,
set conditions, ...)
- Add functional behavior to SQL
 - BEGIN, END, IF, RETURN, ELSE

Programmatic SQL (8)

- Very verbose - specific
- Microsoft has T-SQL
(Transact-SQL)

Programmatic SQL (1)

Python

$x = 2$

① x is an int

② x has value 2

T-SQL

```
DEFINE @x int;  
SET @x = 2;
```



Programmatic SQL (10)

Exciting and new things:

- Transactions
- Stored Procedures
- Triggers

Transactions ()



Transactions (2)

By nature, RDBS are:

- Multi-user
- Concurrent
- Networked

Transactions (J)

Goal : "Freeze" order of actions
until COMMIT or
ROLLBACK

Break



Back at 16:30

Stored Procedures

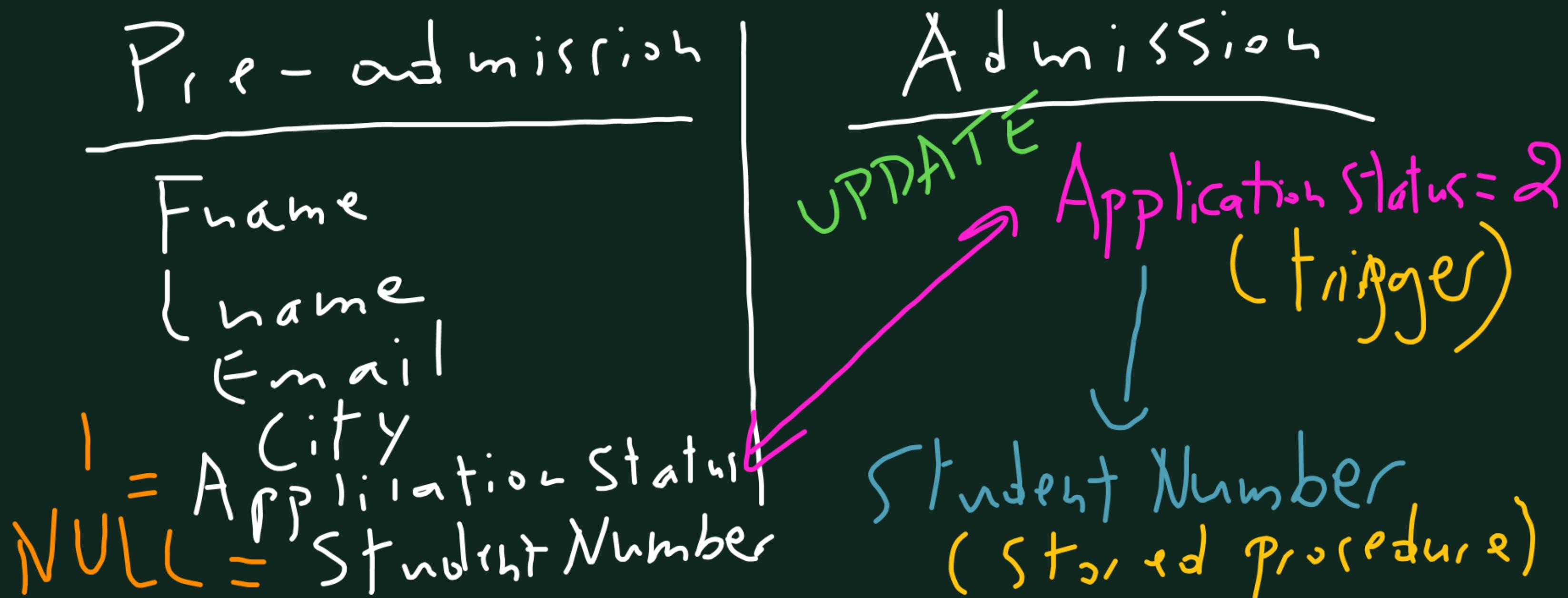
- Reusable pieces of code
- Closest things to actual functions in programming
 - Reusable
 - parametrizable
 - literals

Triggers

fires

Mechanism that executes
a stored procedure based
on actions based
CREATE
UPDATE
DELETE
events

Application : XOU! (1)



Application: You! (2)

Applicant	Student	Student Number
Applicant 1 ✓	1	2024 0001
Applicant 2 ✗	X	4 4
Applicant 3 ✓	2	2024 0002